Exp. No: 3

Date : **VIEWS, SYNONYMS, SEQUENCE, INDEXES, SAVE POINT**

VIEWS :

A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access. Views are known as logical tables. They represent the data of one of more tables.

You can Query, Insert, Update and delete from views, just as any other table.

Syntax :

```
CREATE or REPLACE view view_name AS SELECT column_name(s) FROM table_name
WHERE condition
```

For example ,

// To Create View

```
SQL> CREATE or REPLACE view sale_view as select * from Sale where customer =
'Alex';
SQL>create view emp_det as select e.empno,e.ename,e.sal,e.deptno,d.dname,d.loc from emp e,
dept d where e.deptno=d.deptno;
```

// To display view

```
        SQL> SELECT * from sale_view;
        SQL> SELECT * from emp_det;
```

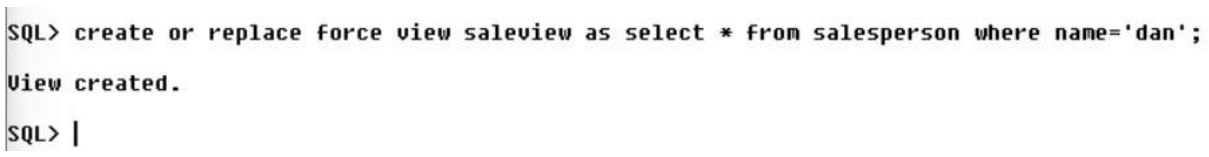
Force View Creation :

This keyword force to create View even if the table does not exist. After creating a force View if we create the base table and enter values in it, the view will be automatically updated.

For Example ,

```
SQL> CREATE or REPLACE force view sale_view1 as select * from Sale1 where customer
= 'John';
```

Output :



```
SQL> create or replace force view saleview as select * from salesperson where name='dan';

View created.

SQL> |
```

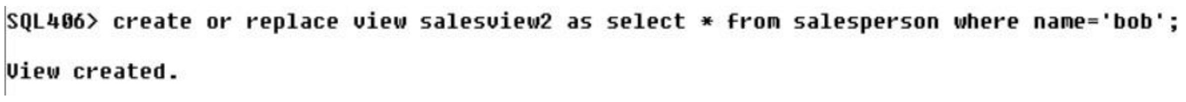
Read-Only View :

We can create a view with read-only option to restrict access to the view.

For Example .

```
      SQL> CREATE or REPLACE view sale_view2 as select * from Sale where customer =
'RAJA' with read-only;
```

Output :

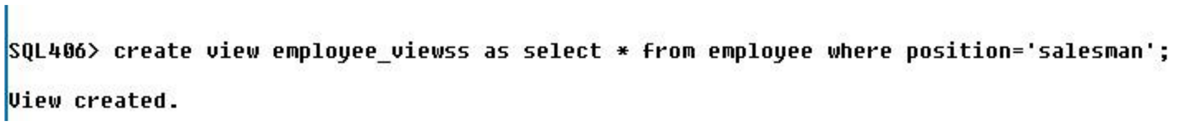


```
SQL406> create or replace view salesview2 as select * from salesperson where name='bob';

View created.
```

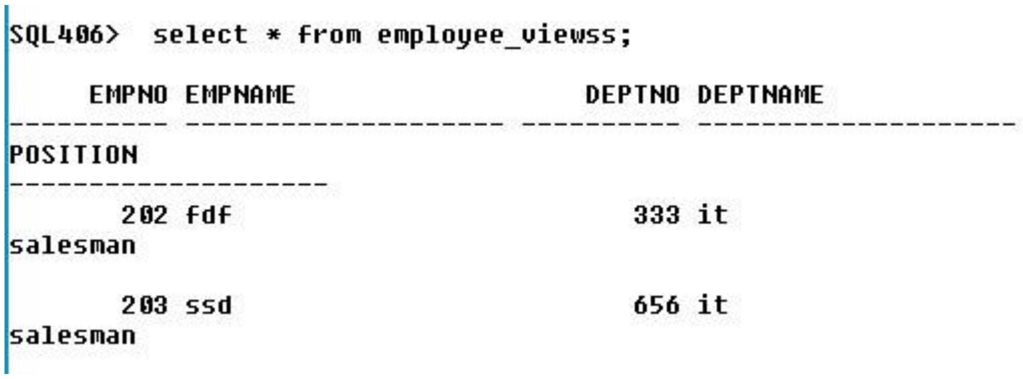
Problem 1: The organization wants to display only the details of the employees those who are slaesman.(horizontal portioning)

Answer:
SQL406> create view employee_viewss as select * from employee where position='salesman';
SQL406> select * from employee_viewss;

SQL> select * from empview1 ; **Sample**
Output :



```
SQL406> create view employee_viewss as select * from employee where position='salesman';

View created.
```



```
SQL406>  select * from employee_viewss;

     EMPNO EMPNAME                  DEPTNO DEPTNAME
---------- -------------------- ---------- --------------------
POSITION
--------------------
       202 fdf                         333 it
salesman

       203 ssd                         656 it
salesman
```

Problem 1: Create the following tables with the mapping given below:
emp_details (emp_no, emp_name, DOB, address, doj, mobile_no, dept_no, salary).

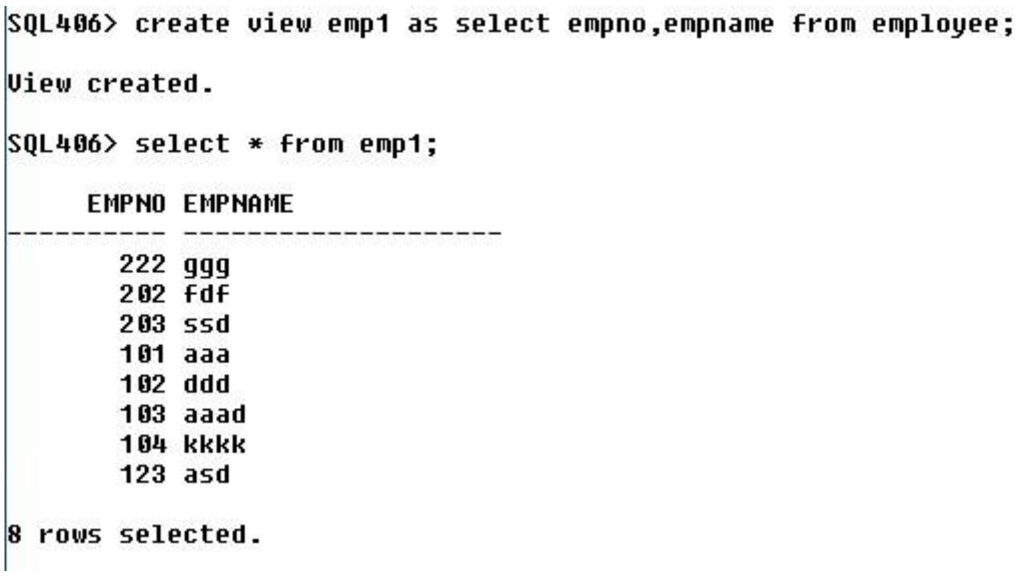
i. Create a view emp1 from emp_details such that it contains only emp_no,emp_name and address

Answer:

SQL406> create view emp1 as select empno,empname from employee;

SQL406> select * from emp1;

Output :



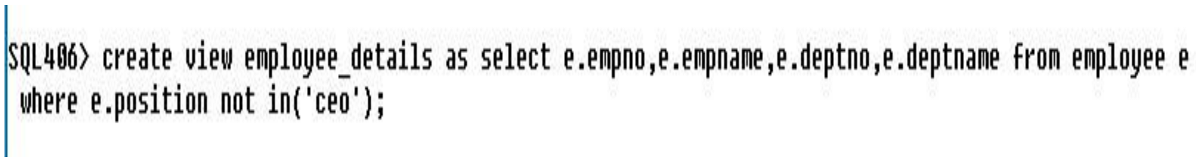
```
SQL406> create view emp1 as select empno,empname from employee;

View created.

SQL406> select * from emp1;

     EMPNO EMPNAME
---------- --------------------
       222 ggg
       202 fdf
       203 ssd
       101 aaa
       102 ddd
       103 aaad
       104 kkkk
       123 asd

8 rows selected.
```

Problem 2: The organization wants to display only the details like empno,empname,deptno,dname of the all the employees except the CEO . (full portioning) Answer:

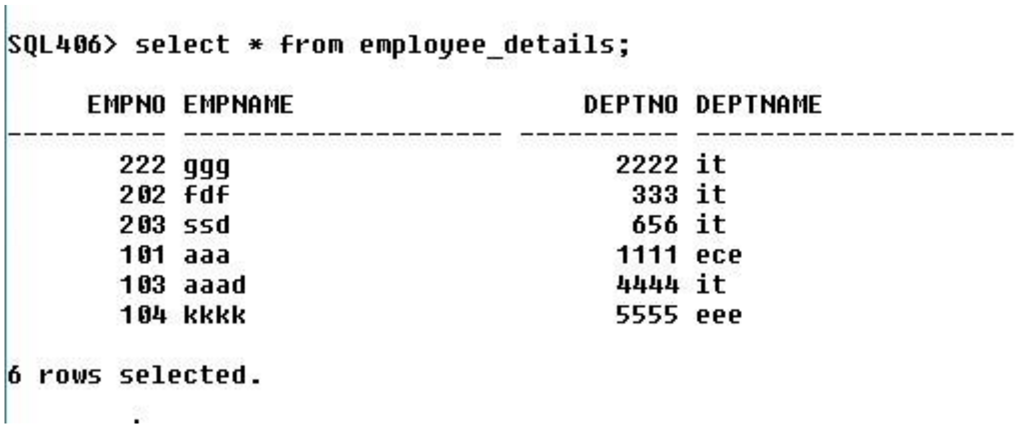
SQL406> create view employee_details as select e.empno,e.empname,e.deptno,e.deptname from employee e where e.position not in('ceo');

SQL406> select * from employee_details;

Sample Output :



```
SQL406> create view employee_details as select e.empno,e.empname,e.deptno,e.deptname from employee e
 where e.position not in('ceo');
```

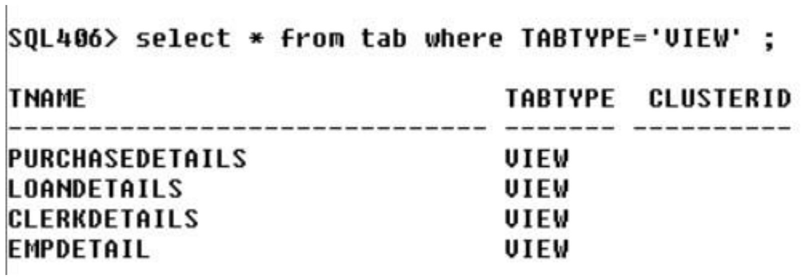


```
SQL406> select * from employee_details;

     EMPNO EMPNAME                  DEPTNO DEPTNAME
---------- -------------------- ---------- --------------------
       222 ggg                        2222 it
       202 fdf                         333 it
       203 ssd                         656 it
       101 aaa                        1111 ece
       103 aaad                       4444 it
       104 kkkk                       5555 eee

6 rows selected.
```

Problem 3: Display all the views generated.

Answer: select * from tab where TABTYPE='VIEW' ;

Sample Output :

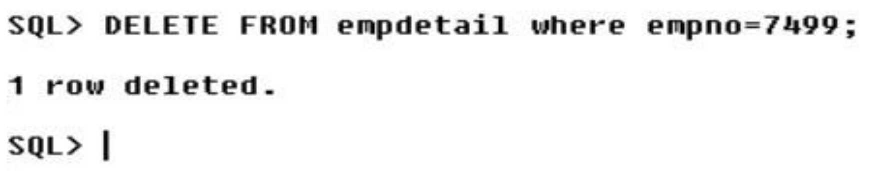


```
SQL406> select * from tab where TABTYPE='VIEW' ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
PURCHASEDETAILS                VIEW
LOANDETAILS                    VIEW
CLERKDETAILS                   VIEW
EMPDETAIL                      VIEW
```

Problem 4: Execute the DML commands delete to delete a record from the view created.

Answer : SQL406> DELETE FROM empdetail where empno=7499;

Sample Output :

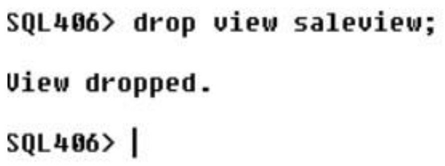


```
SQL> DELETE FROM empdetail where empno=7499;

1 row deleted.

SQL> |
```

Problem 5: Drop a view.

Answer: SQL406> drop view saleview;

Sample Output :



```
SQL406> drop view saleview;

View dropped.

SQL406> |
```

SEQUENCES :

A sequence is used to generate numbers in sequence. **Syntax :**

CREATE Sequence sequence-name start with initial-value increment by increment-value maxvalue maximum-value cycle | nocycle

Problem 1 : Create a sequence name as "bills" with following constraints like start with 1 , Minimum value is 1 and Maximum Value is 100 ;

Answer : create sequence **bills** start with 1 increment by 1 **minvalue** 1 **maxvalue** 100 cycle cache 10;

Sample Output :

Sequence created.

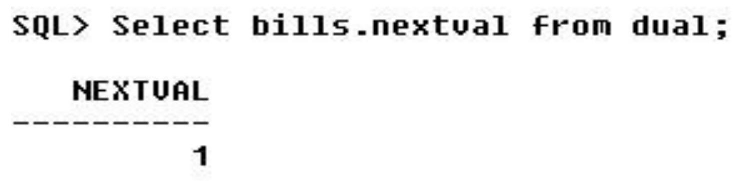
Accessing Sequence Numbers :

To generate Sequence Numbers you can use NEXTVAL and CURRVAL.

Problem 2 : Select and display the next value of sequence generated named as "bills".

Answer : SQL> Select **bills.nextval** from dual;

Sample Output :

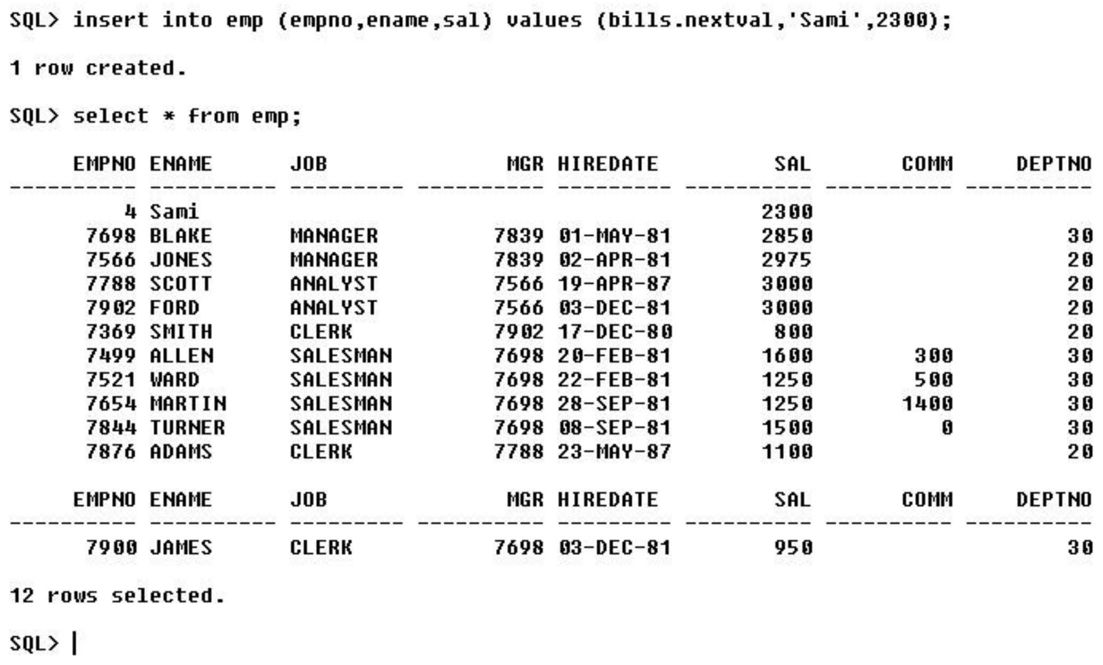


```
SQL> Select bills.nextval from dual;

   NEXTVAL
----------
         1
```

Problem 3 : Insert the next value of the "bills" sequence into emp table empno column .

Answer : SQL> insert into emp (empno,ename,sal) values (**bills.nextval**,'Sami',2300);

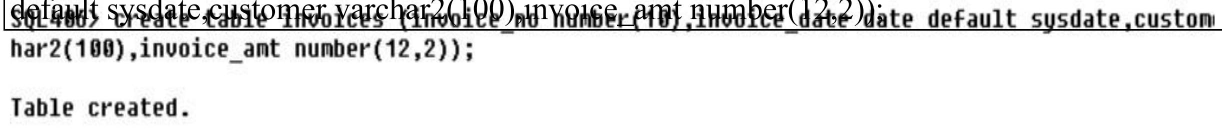
Sample Output :



```
SQL> insert into emp (empno,ename,sal) values (bills.nextval,'Sami',2300);

1 row created.

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
         4 Sami                                            2300
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30

12 rows selected.

SQL> |
```

Problem 4 : Creating Table with sequences and default :

Answer : create table invoices (invoice_no number(10) default bills.nextval, invoice_date date default sysdate,customer varchar2(100),invoice_amt number(12,2));



```
SQL400> create table invoices (invoice_no number(10),invoice_date date default sysdate,custom
har2(100),invoice_amt number(12,2));

Table created.
```

Table Created

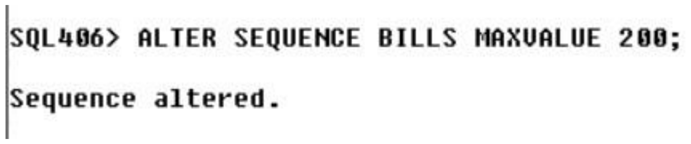
Altering Sequences :

To alter sequences use ALTER SEQUENCE statement.

Problem 5 : Alter the sequence named as "bills" for update maximum values is 200

```
SQL> ALTER SEQUENCE BILLS MAXVALUE 200;
```

Sample output :



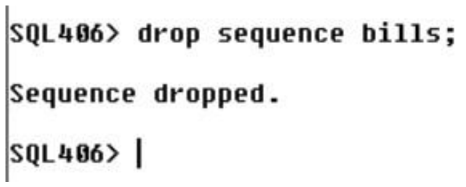
```
SQL406> ALTER SEQUENCE BILLS MAXVALUE 200;

Sequence altered.
```

Sequence altered.

Dropping Sequences :

```
SQL> drop sequence bills;
```

Sample output :

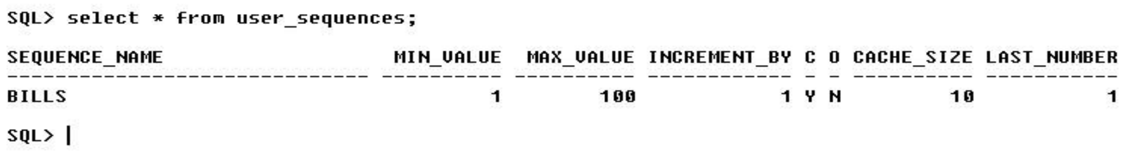


```
SQL406> drop sequence bills;

Sequence dropped.

SQL406> |
```

Sequence dropped.

Listing Information About Sequences :

SQL> select * from user_sequences;



```
SQL> select * from user_sequences;

SEQUENCE_NAME                   MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------------------------ ---------- ---------- ------------ - - ---------- -----------
BILLS                                   1        100            1 Y N         10           1

SQL> |
```

SYNONYMS :

A synonym is an alias or alternative name for objects like a table, view, snapshot, sequence, procedure, function, or package.

Two types of SYNONYMS are,

Public Synonym

Private Synonym **Syntax :**

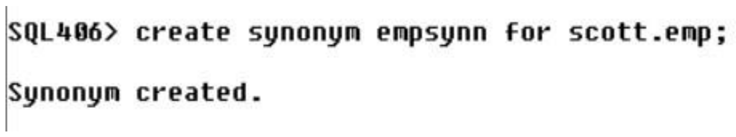
```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym_name FOR [schema
.] object_name [@ dblink];
```

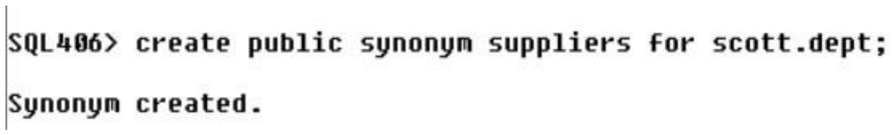
Problem 1: Create the synonyms for any object like table emp ,

```
SQL> create synonym empsyn for scott.emp;
```

SQL>create **public** synonym suppliers for scott.dept; // accessible to all users

Sample Output :


```
SQL406> create synonym empsynn for scott.emp;

Synonym created.
```

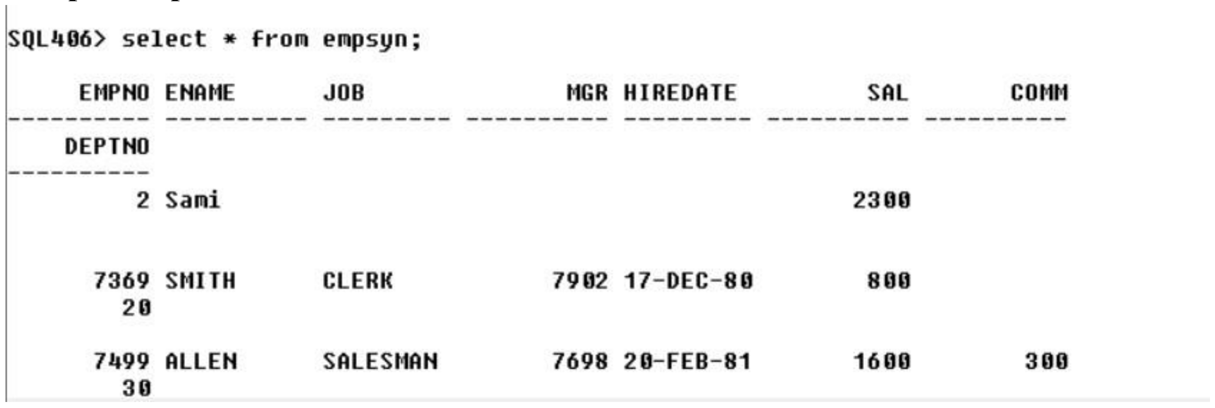

```
SQL406> create public synonym suppliers for scott.dept;

Synonym created.
```

Synonym created.

Problem 2: View the Synonym :

SQL> select * from employee;

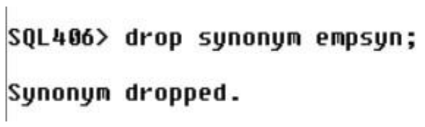
Sample Output :


```
SQL406> select * from empsyn;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
         2 Sami                                            2300


      7369 SMITH      CLERK           7902 17-DEC-80        800
        20

      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300
        30
```

Problem 3: Dropping Synonyms :

SQL> drop synonym employee;

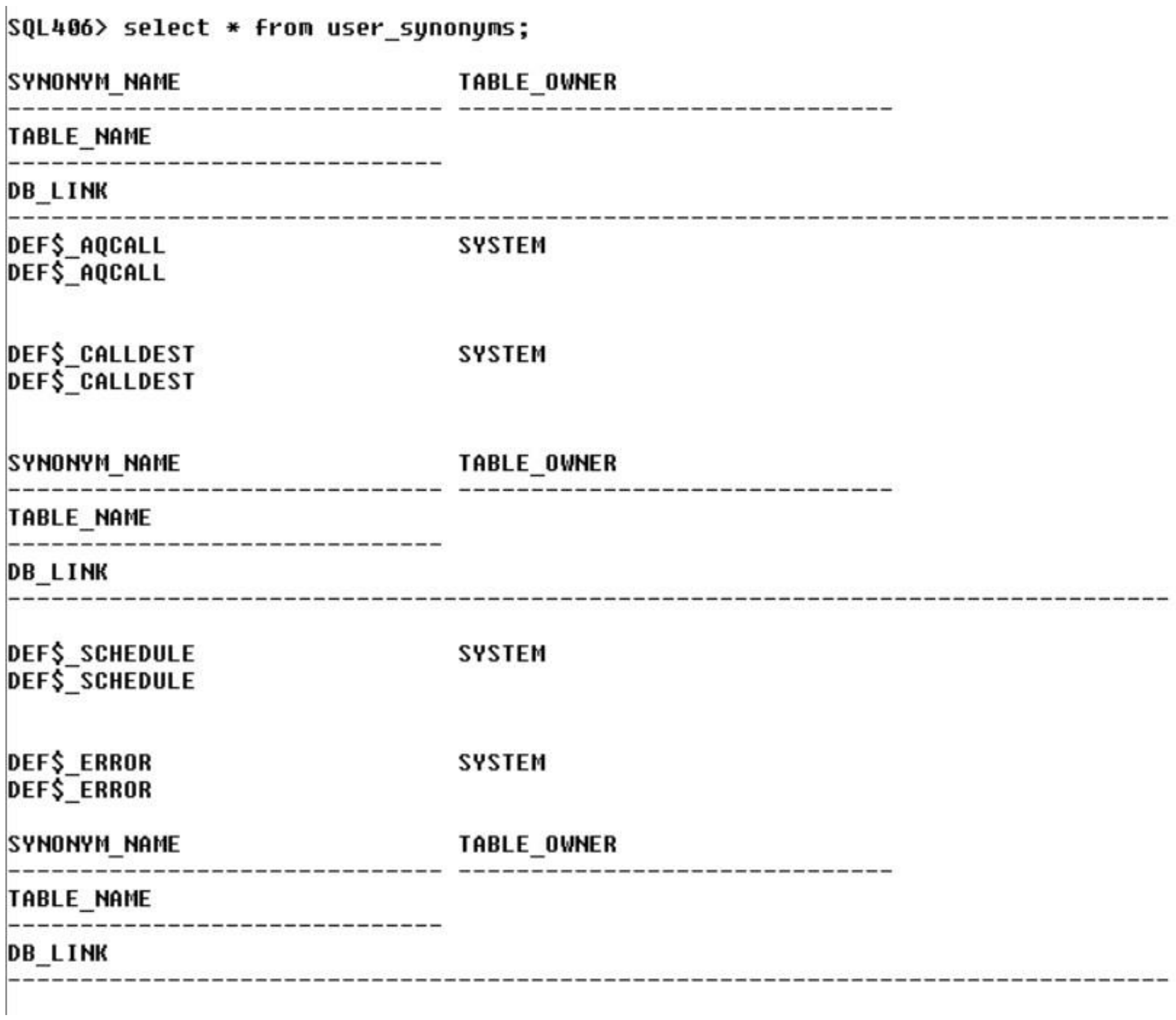
Sample Output :


```
SQL406> drop synonym empsyn;

Synonym dropped.
```

Synonym Dropped

Problem 4: Listing information about synonyms

SQL> select * from user_synonyms;



```
SQL406> select * from user_synonyms;

SYNONYM_NAME                   TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME
------------------------------
DB_LINK
--------------------------------------------------------------------------------
DEF$_AQCALL                    SYSTEM
DEF$_AQCALL


DEF$_CALLDEST                  SYSTEM
DEF$_CALLDEST


SYNONYM_NAME                   TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME
------------------------------
DB_LINK
--------------------------------------------------------------------------------

DEF$_SCHEDULE                  SYSTEM
DEF$_SCHEDULE


DEF$_ERROR                     SYSTEM
DEF$_ERROR

SYNONYM_NAME                   TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME
------------------------------
DB_LINK
--------------------------------------------------------------------------------
```

INDEX :

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. By default, Oracle creates B-tree indexes.

Syntax :

```
        CREATE INDEX index_name ON table_name (column_name);
```

For Example ,

SQL> CREATE INDEX empIndex ON emp(LastName);   // Index on Single Column
SQL> CREATE INDEX supplier_idx ON supplier (supplier_name, city);   // Index on Multiple Column
SQL>create index empno_ind on emp (empno);   **SQL>**create index empdept_ind on emp (empno,deptno);

Rename an Index :

Syntax :

```
ALTER INDEX index_name   RENAME TO new_index_name;
```

For Example ,

```
SQL> ALTER INDEX supplier_idx RENAME TO supplier_index_name;
```

Drop an Index :

Syntax :

```
DROP INDEX index_name;
```

For example:

```
SQL>  DROP INDEX supplier_idx;
SQL>select * from user_indexes;
```

Problem 3.1:

Consider the following relational schema for a Sales database application:

Product (Prodid, Prodesc, Price, Stock)

Purchase (Purid, Proid, qty, supplierName)

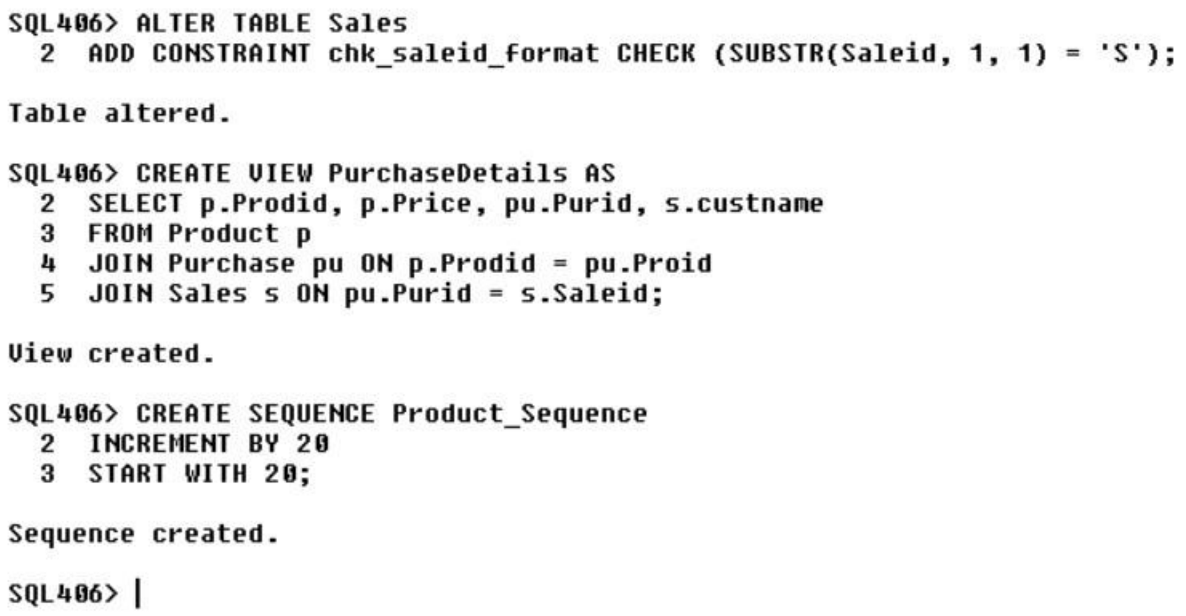
Sales (Saleid, Proid, qty, custname)

a. **Include the constraint on Saleid that it starts with letter ‘S’.**
b. **Create a view that keeps track of Prodid, price, Purid and customerName who made the purchase.**
c. **Create a sequence named Product_Sequence that gets incremented by 20 and use it for inserting Prodid values in Product table.**

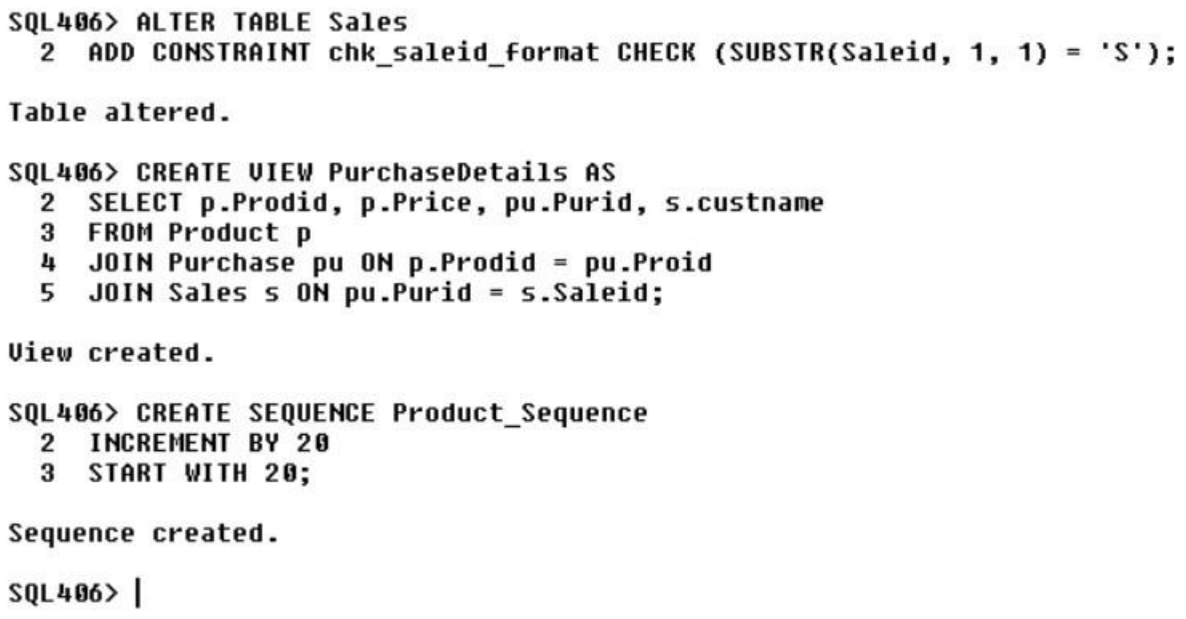
Output:

Answer :

```
CREATE TABLE Customer (Custid INT PRIMARY KEY, Custname VARCHAR2(100), Addr
VARCHAR2(255),phno VARCHAR2(20), panno VARCHAR2(20) );

CREATE TABLE Loan ( Loanid INT PRIMARY KEY, Amount DECIMAL(10, 2), Interest DECIMAL(5, 2),Custid
INT, FOREIGN KEY (Custid) REFERENCES Customer(Custid));

CREATE TABLE Account (Accid INT PRIMARY KEY,Accbal DECIMAL(10, 2),Custid INT,FOREIGN KEY(Custid)
REFERENCES Customer(Custid));

ALTER TABLE SalesADD CONSTRAINT chk_saleid_format CHECK (SUBSTR(Saleid, 1, 1) = 'S');

CREATE VIEW PurchaseDetails AS SELECT p.Prodid, p.Price, pu.Purid, s.custname FROM Product p JOIN
Purchase pu ON p.Prodid = pu.Proid JOIN Sales s ON pu.Purid = s.Saleid;

CREATE SEQUENCE Product_SequenceINCREMENT BY 20START WITH 20;
```



```
SQL406> ALTER TABLE Sales
  2  ADD CONSTRAINT chk_saleid_format CHECK (SUBSTR(Saleid, 1, 1) = 'S');

Table altered.

SQL406> CREATE VIEW PurchaseDetails AS
  2  SELECT p.Prodid, p.Price, pu.Purid, s.custname
  3  FROM Product p
  4  JOIN Purchase pu ON p.Prodid = pu.Proid
  5  JOIN Sales s ON pu.Purid = s.Saleid;

View created.

SQL406> CREATE SEQUENCE Product_Sequence
  2  INCREMENT BY 20
  3  START WITH 20;

Sequence created.

SQL406> |
```



```
SQL406> ALTER TABLE Sales
  2  ADD CONSTRAINT chk_saleid_format CHECK (SUBSTR(Saleid, 1, 1) = 'S');

Table altered.

SQL406> CREATE VIEW PurchaseDetails AS
  2  SELECT p.Prodid, p.Price, pu.Purid, s.custname
  3  FROM Product p
  4  JOIN Purchase pu ON p.Prodid = pu.Proid
  5  JOIN Sales s ON pu.Purid = s.Saleid;

View created.

SQL406> CREATE SEQUENCE Product_Sequence
  2  INCREMENT BY 20
  3  START WITH 20;

Sequence created.

SQL406> |
```

Problem 3.2 :

i.Consider the following relational schema for a Loan database application:

Customer (Custid, Custname, Age, phno)

Loan (Loanid, Amount, Custid)

a. Create the above mentioned tables .

b. Include the constraint on Loanid that it starts with letter ‘Lo’.

c. Create a view that keeps track of Custid, Custname, loanid and loan amount.

Answer :

CREATE TABLE Customer (Custid INT PRIMARY KEY, Custname VARCHAR(255),Age

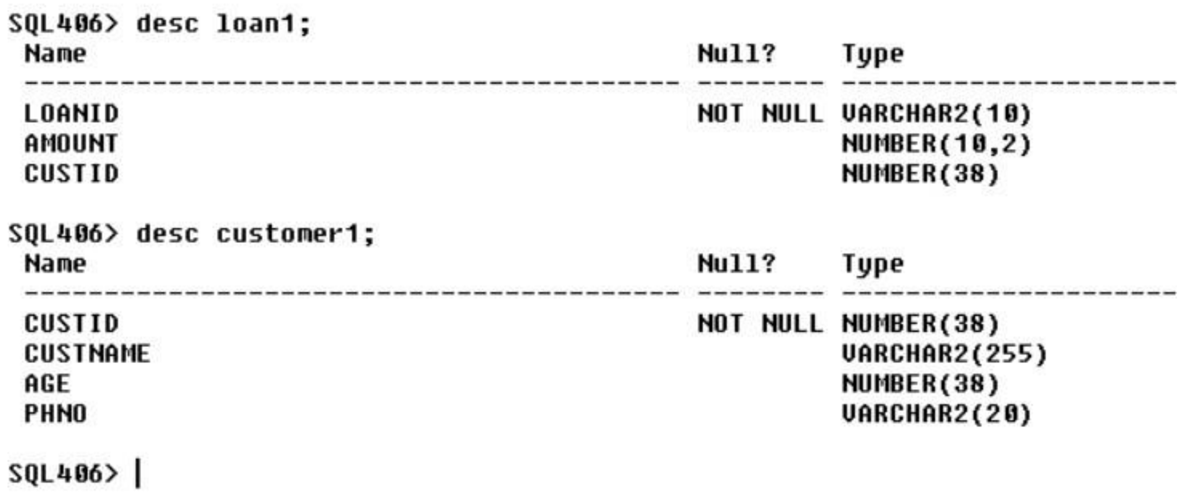
INT, phno VARCHAR(20));

CREATE TABLE Loan (Loanid VARCHAR(10) PRIMARY KEY, Amount DECIMAL(10,
2), Custid INT, FOREIGN KEY (Custid) REFERENCES Customer(Custid));

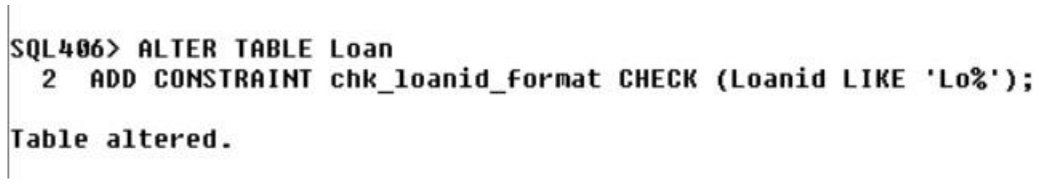
ALTER TABLE Loan ADD CONSTRAINT chk_loanid_format CHECK (Loanid LIKE
'Lo%');

CREATE VIEW LoanDetails AS SELECT c.Custid, c.Custname, l.Loanid, l.Amount FROM
Customer c JOIN Loan l ON c.Custid = l.Custid;

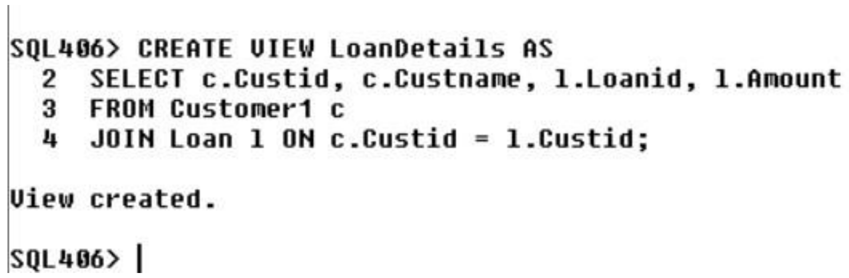
Output :



```
SQL406> desc loan1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOANID                                    NOT NULL VARCHAR2(10)
 AMOUNT                                             NUMBER(10,2)
 CUSTID                                             NUMBER(38)

SQL406> desc customer1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTID                                    NOT NULL NUMBER(38)
 CUSTNAME                                           VARCHAR2(255)
 AGE                                                NUMBER(38)
 PHNO                                               VARCHAR2(20)

SQL406> |
```



```
SQL406> ALTER TABLE Loan
  2  ADD CONSTRAINT chk_loanid_format CHECK (Loanid LIKE 'Lo%');

Table altered.
```



```
SQL406> CREATE VIEW LoanDetails AS
  2  SELECT c.Custid, c.Custname, l.Loanid, l.Amount
  3  FROM Customer1 c
  4  JOIN Loan l ON c.Custid = l.Custid;

View created.

SQL406> |
```

Problem 3.3 :

Consider the following employee and department tables:
EMPLOYEE(empno, ename, designation, manager, hiredate, salary, commission, deptno)
DEPARTMENT(deptno, dname, location)

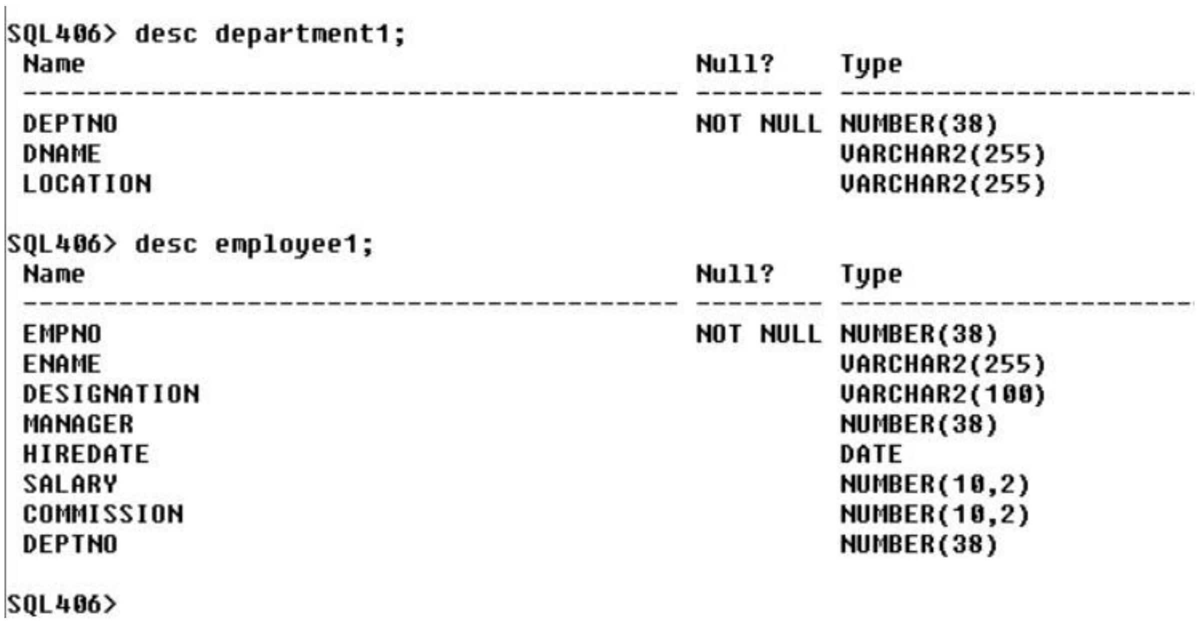
i. **Create a view which consists of details of all 'CLERK'**

Answer :

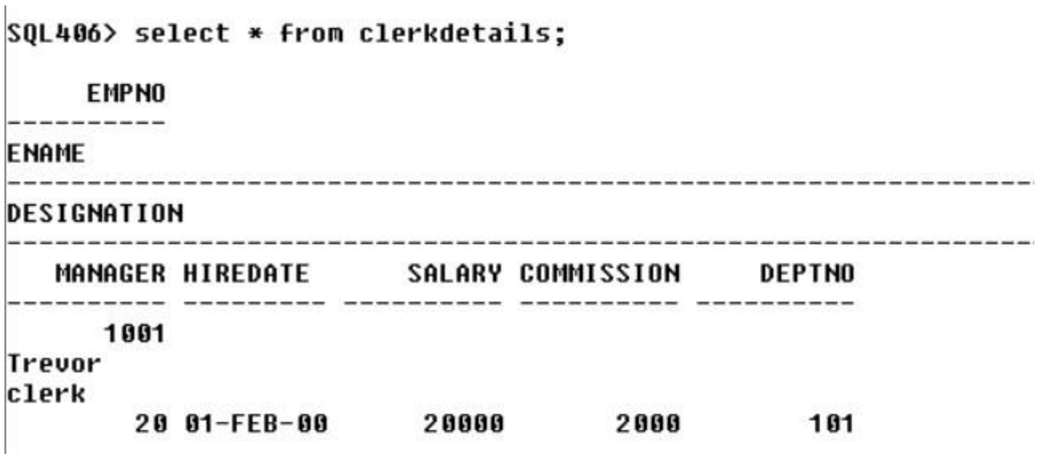
```
SQL406> CREATE TABLE DEPARTMENT1 (  deptno INT PRIMARY KEY,dname
VARCHAR(255),location VARCHAR(255));

SQL406> CREATE TABLE EMPLOYEE1 (empno INT PRIMARY KEY ,
ename VARCHAR(255),       designation VARCHAR(100),   manager INT,   hiredate DATE,
salary DECIMAL(10, 2),       commission DECIMAL(10, 2),   deptno INT,  FOREIGN KEY
(deptno) REFERENCES DEPARTMENT1(deptno));
```

Output :

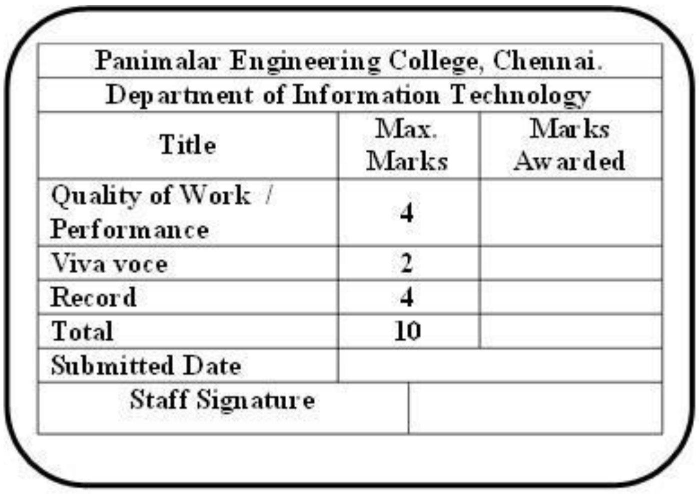
```
SQL406> CREATE VIEW ClerkDetails AS SELECT empno, ename, designation, manager,
hiredate, salary, commission, deptno            FROM EMPLOYEE1 WHERE designation =
'clerk'; SQL406> select * from clerkdetails;
```



```
SQL406> desc department1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(38)
 DNAME                                              VARCHAR2(255)
 LOCATION                                           VARCHAR2(255)

SQL406> desc employee1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(38)
 ENAME                                              VARCHAR2(255)
 DESIGNATION                                        VARCHAR2(100)
 MANAGER                                            NUMBER(38)
 HIREDATE                                           DATE
 SALARY                                             NUMBER(10,2)
 COMMISSION                                         NUMBER(10,2)
 DEPTNO                                             NUMBER(38)

SQL406>
```



```
SQL406> select * from clerkdetails;

     EMPNO
----------
ENAME
--------------------------------------------------------------------------------
DESIGNATION
--------------------------------------------------------------------------------
   MANAGER HIREDATE      SALARY COMMISSION     DEPTNO
---------- --------- ---------- ---------- ----------
      1001
Trevor
clerk
        20 01-FEB-00      20000       2000        101
```



Panimalar Engineering College, Chennai.		
Department of Information Technology		
Title	Max. Marks	Marks Awarded
Quality of Work / Performance	4	
Viva voce	2	
Record	4	
Total	10	
Submitted Date		
Staff Signature		

Result :

The implementation of Views, Synonyms, Sequence, Indexes, Save Point was successfully done and verified.